NASA/TM—2001-211154

Survey of Header Compression Techniques

Joseph A. Ishac
Glenn Research Center, Cleveland, Ohio

September 2001

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized data bases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/TM—2001-211154

Survey of Header Compression Techniques

Joseph A. Ishac
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center

September 2001

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov/GLTRS

Table of Contents

Survey of Header Compression Techniques

Joseph Ishac
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Summary

This report provides a summary of several different header compression techniques. The different techniques included are:

- Van Jacobson's header compression [RFC1144]
- SCPS header compression [SCPS-TP, SCPS-NP]
- Robust header compression [ROHC]
- The header compression techniques in [RFC2507] and [RFC2508]

The methodology for compression and error correction for these schemes are described in the remainder of this document. All of the header compression schemes support compression over simplex links, provided that the end receiver has some means of sending data back to the sender. However, if that return path does not exist, then neither Van Jacobson's nor SCPS can be used, since both rely on TCP. In addition, under link conditions of low delay and low error, all of the schemes perform as expected. However, based on the methodology of the schemes, each scheme is likely to behave differently as conditions degrade. Van Jacobson's header compression relies heavily on the TCP retransmission timer and would suffer an increase in loss propagation should the link possess a high delay and/or bit error rate (BER). The SCPS header compression scheme protects against high delay environments by avoiding delta encoding between packets. Thus, loss propagation is avoided. However, SCPS is still affected by an increased BER since the lack of delta encoding results in larger header sizes. Next, the schemes found in [RFC2507] and [RFC2508] perform well for non-TCP connections in poor conditions. [RFC2507] performance with TCP connections is improved by various techniques over Van Jacobson's, but still suffers a performance hit with poor link properties. Also, [RFC2507] offers the ability to send TCP data without delta encoding, similar to what SCPS offers. ROHC is similar to the previous two schemes, but adds additional CRC's into headers and improves compression schemes which provide better tolerances in conditions with a high BER.

Introduction

This report provides a summary of several different header compression techniques currently in use as well as a few which are still conceptual.

1 The TCP/IP Header

In order to understand the compression techniques used by the various header compression algorithms described in this paper, it is necessary to reference the layout of a standard TCP/IP header. Figure 1 shows a standard TCP/IP header as described in [RFC793] and [RFC791] respectively. A minimal sized header consists of all the fields shown below save for any options. The resulting 40-byte header is indexed at the left of the figure.

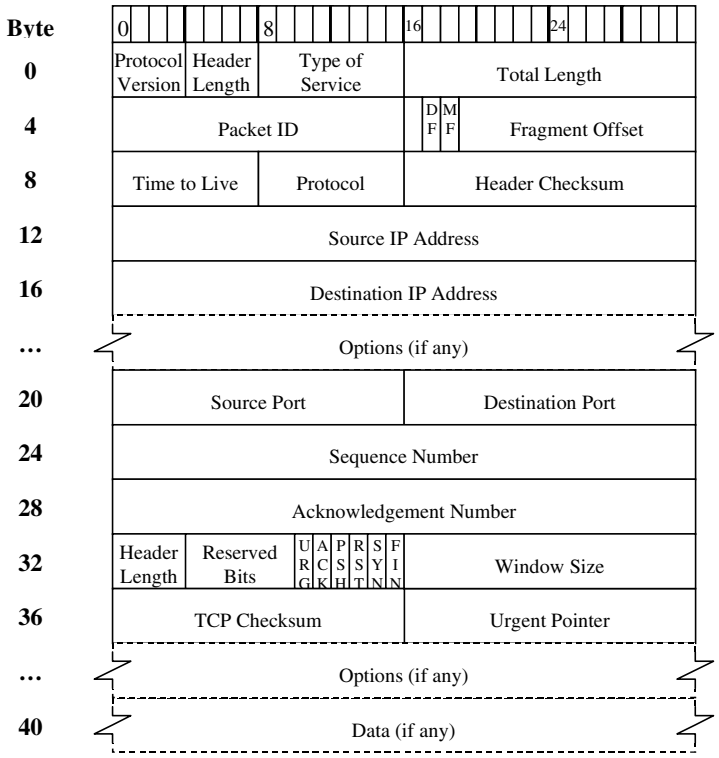

Figure 1: Format of a TCP/IP Header

2 Van Jacobson's Header Compression (RFC 1144)

The main goal behind [RFC1144] header compression scheme is to improve the line efficiency[1] for a serial link. For example, such improvements are beneficial in interactive connections in which a typed character results in a 41-byte TCP/IP packet (40 bytes for the header and one byte for the character). Increasing the line efficiency also allows for better allocation in asymmetric bandwidth allocation schemes. Since allocation is often

[1] Line efficiency is defined as the ratio of data to total bytes transferred over a link.

dependent on the amount of data to be sent or received, the smaller headers will cause less fluctuation in the amount of actual data traversing the link. Also for this particular header compression scheme, the TCP and IP headers are compressed together and not individually. Compressing each header separately decreases the performance of both the compression ratio and the efficiency of the (de)compression code [RFC1144].

2.1 Compressed Header

Compression in [RFC1144] relies on the fact that for a particular TCP/IP connection, over 50% of the fields within the header remain constant. Figure 2 shows the fields of interest and which often change for a minimal standard header. For the IP portion of the header only the *packet id* is of concern as the *total length* is expected to be handled by the framing protocol. The IP *header checksum* is also unnecessary, as most of the IP header is not being transmitted.

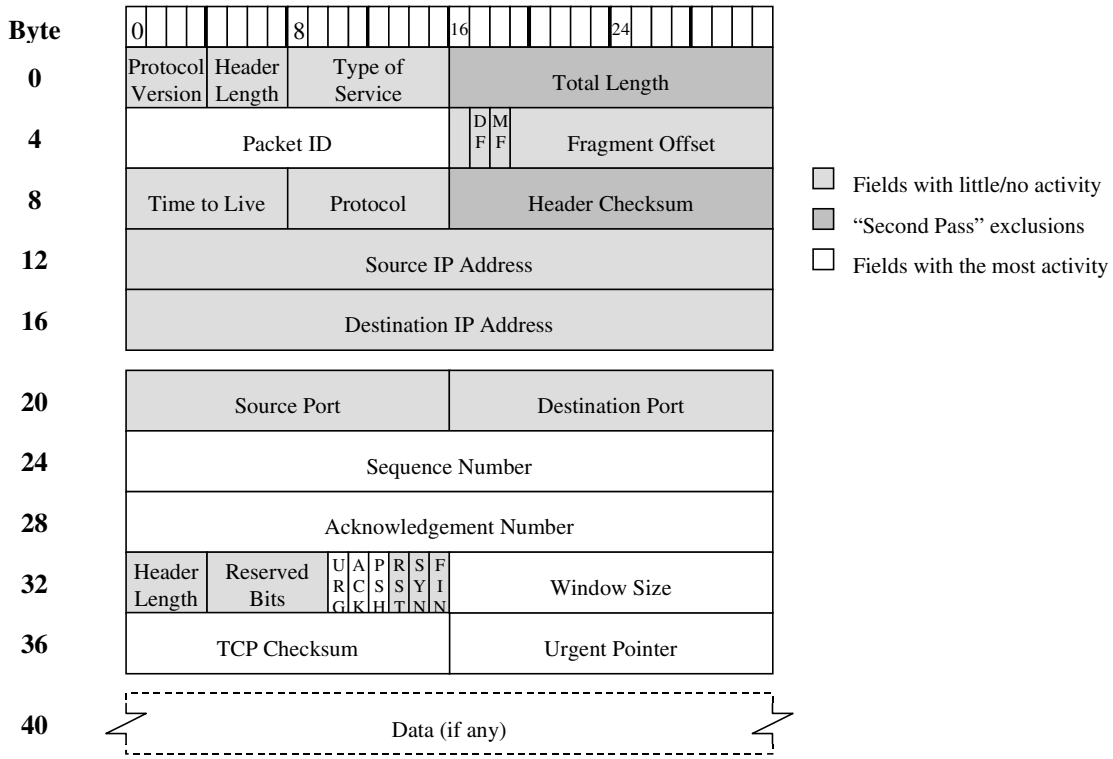

Figure 2: Active fields within a TCP/IP Header [RFC1144]

While certain fields may change, they often do not all change at once. Thus, values which do not change can be omitted from the compressed header. Also, the change in values between packets for any particular field is often smaller than the field's actual value and thus can be represented in a smaller amount of bytes. Figure 3 shows the header of a compressed TCP/IP segment.

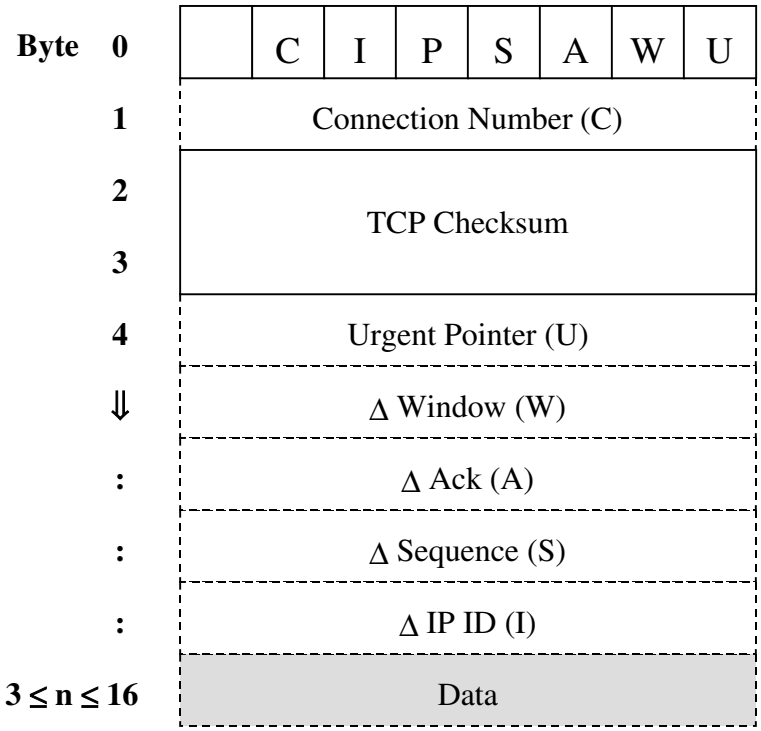

Figure 3: Compressed TCP/IP Header [RFC1144]

The first byte of the compressed packet represents a bit mask which indicates which of the optional (dashed) fields are present within the header. If a field is not present its value is presumed to be zero.[2] The only value in the mask which does not follow this rule is the *push* (P) bit, which is an exact copy of the push bit within the uncompressed packet. The *connection number* (C) allows the (de)compressor to locate the last saved header for a particular connection.[3] If the *connection number* is unchanged from the previously compressed packet it can be omitted. By using the stored headers, differences can be utilized instead of actual values. Changes of one to 255 in value are represented by a single byte. Since zero values are not sent, a change of zero indicates that the next two bytes represent the MSB and LSB of a 16 bit value. If the difference should be negative or more than 64K an uncompressed packet is sent.[4] The TCP checksum is sent unmodified from the uncompressed packet in order to preserve the end-to-end integrity of the data. Finally, it should be noted that a pair of special cases exist which make use of the rare combination of S, A, W, and U bits in order to handle common terminal type traffic. Since the condition where the S, W, and U bits are all set simultaneously is rare, it aids in further reducing the amount of header needed for those types of traffic. Should such a condition occur naturally, an uncompressed packet is set.

[2] There is an exception. If *packetID* (I) is clear, the assumed difference is one, not zero.
[3] Uniquely identified by the addresses and ports of the source and destination.
[4] With the exception of the change in window size, which can be both positive and negative. Such negative changes are handled with two's complement arithmetic. Also, the connection number is limited to a single byte and cannot be extended (allowing for 256 simultaneously active TCP connections).

2.2 Error Correction and Recovery

The decompressor relies on two forms of error detection in order to capture corrupted packets. The first is the reception of CRC errors from the framer for a particular packet. The second is the failure of the TCP checksum at the transport layer. Thus no error checking is actually done within the decompressor. CRC errors must be reported by the framer in order to prevent an undetected change in connection number.[5] Thus, if a CRC error is reported, packets are discarded until a connection number is explicitly received. In the case of TCP checksum errors, there are two cases that need to be handled. The first is the occurrence of a checksum error in the forward path, or where data packets suffer checksum errors after decompression. As a result of the errors, the receiver does not send any acknowledgements to the sender. Without acknowledgements, the sender eventually suffers a timeout (RTO) and retransmits. However, the retransmission carries a sequence number with a negative difference to the last packet sent by the sender and is sent uncompressed. Thus, the state of the decompressor at the TCP receiver is corrected. A similar situation occurs in the reverse path, or where acknowledgements returning to the TCP sender contain checksum errors. In this case the meaningless acknowledgements cause the sender to timeout and retransmit data. The retransmissions cause the receiver to generate duplicate acknowledgements, which carry the same properties as the last packet and are sent uncompressed. Thus, the state of the decompressor at the TCP sender is corrected.

2.3 Compressor Location and Efficiency

The compressor is located between the network (IP) and framing layers, and relies on the framer to provide in-order packet delivery with good error detection. No feedback takes place between the compressor and decompressor. State must also be kept for each active TCP/IP connection, with each connection receiving its own connection number. Since this stored header is needed for calculating differences at both the compressor and decompressor, a successful transmission of an uncompressed packet is needed to (re-)synchronize the decompressor. Also, storage of the last header allows for very quick manipulations of differences, taking only three instructions on a Motorola 68K family processor.

3 Space Communication Protocol Specification (SCPS)

The SCPS protocol was designed to be "applicable to any kind of space mission or infrastructure, regardless of complexity" [SCPS-NP]. In order to do this, SCPS redefines the network stack from the network layer down with variations to known protocols (i.e. SCPS-NP is used instead of IP). Modifications to TCP are done through use of TCP extensions or options as specified in [SCPS-TP]. This document summarizes the compression scheme available when using both [SCPS-NP] and [SCPS-TP].

[5] For example, take three packets arriving from two connections in the following order: C_1, C_2, C_2. Since the connection number does not change it is omitted from the second C_2. However the first C_2 suffers from a CRC error. If that error were not reported, then there is at least a 2^{-16} chance that the TCP receiver accepts the second C_2 as part of C_1.

3.1 Compressed Header

In the SCPS header compression scheme, the transport and network layer headers are dealt with separately. As a result, compression of the SCPS-TP header is negotiated in the initial uncompressed three-way handshake. If both sides agree, indication of the compression is set in the SCPS-NP header. Figure 4 shows the SCPS-NP header with the size of each field indicated in bits. Shaded fields are optional and are set through the *control mask*. The minimum header size is four bytes and is indexed to the left of the figure. Aside from this minimum size, there is no compression scheme for the SCPS-NP header. Since SCPS-TP is an extension of TCP, it does not have its own header. Instead a four byte TCP option is included in the standard TCP header. Figure 5 shows the SCPS-TP option.

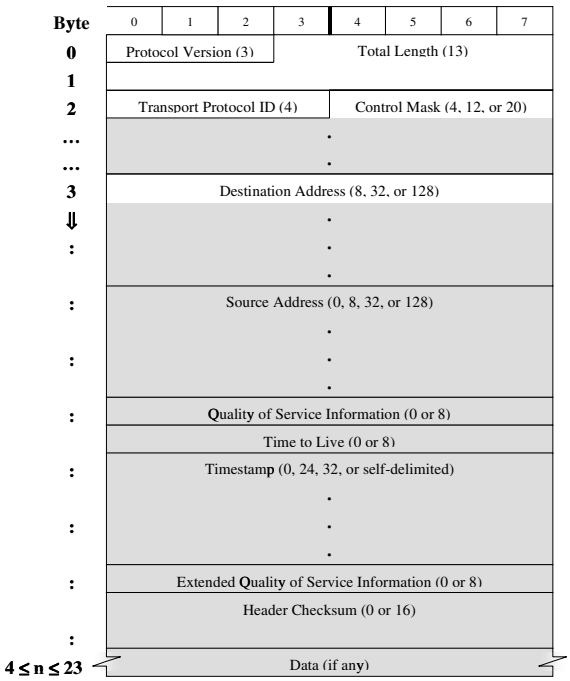

Figure 4: SCPS Network Protocol Header

Byte 1	Option Type = SCPS = decimal 10
Byte 2	Option Length = 4
Byte 3	SCPS Options (Bit-Vector)
Byte 4	Connection ID

Figure 5: SCPS Transport Protocol Option for TCP

The layout of the compressed SCPS-TP header is similar to that found in [RFC1144] and is shown in Figure 6. The first byte of the header is mandatory and contains the *connection ID* from the SCPS option field in order to identify which connection the packet is associated with. The second byte of the header composes a bit-vector which indicates the presence of other fields within the header. By setting the "More" bit, the

bit-vector can be extended to a second byte in order to accommodate less frequent header values. If a field is not included its value is presumed to be zero. Inclusion of the remaining optional fields, which are indicated by dashes, is determined by the compressor. Table 1 summarizes the conditions necessary to include a particular field. Each included field is a copy of their respective uncompressed values in the original TCP header. The *checksum* is also mandatory and is the last element in the header.

Field (bit in bit-vector)	**Condition to send field**
Urgent Pointer (URG)	URG is set in the original header
Window (A) & ACK Number (A)	Acknowledgement number changed from last packet sent or if the current packet is a replica of the previous packet
Sequence Number (S)	The packet contains data or the FIN flag is set in the original header
Outbound Timestamp (TS1)	The TCP Timestamp option is present and the use of network layer timestamps was determined unavailable upon connection setup
Echo Reply Timestamp (TS2)	The TCP Timestamp option is present
TCP Options (Opts)	There are other TCP options which remain
Pad (Pad)	The bit-vector consists of two bytes and the header occupies an odd number of bytes
--- (Push)	The push bit is set in the original header
--- (AckR)	Both the ACK and RST bits are set in the original header
--- (RST)	RST is set in the original header
--- (FIN)	FIN is set in the original header

Table 1: Requirements to send fields in a compressed SCPS-TP header

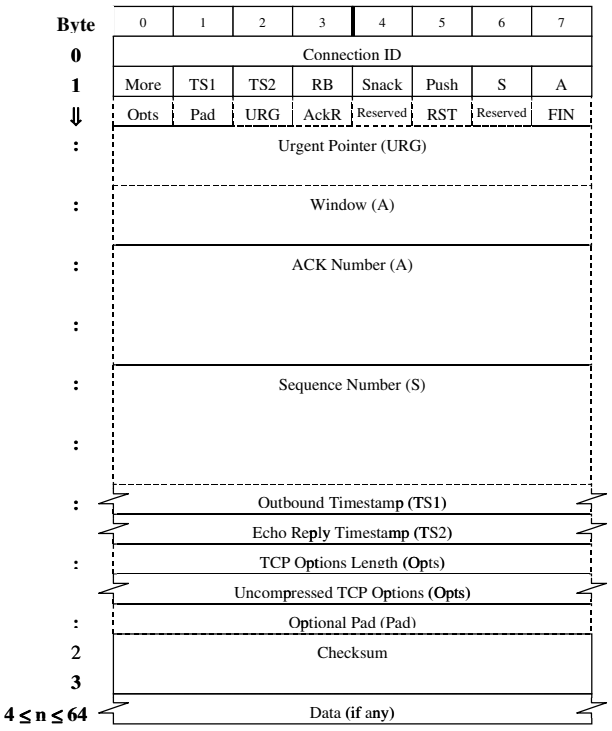

Figure 6: Compressed SCPS-TP Header

3.2 Error Correction and Recovery

Since every field transmitted in the compressed header represents actual values, there is no need for any error correction or recovery on part of a SCPS decompressor.

3.3 Compressor Location and Efficiency

Although the location of a SCPS is not clearly stated, it is more than likely part of transport layer (specifically SCPS-TP). The reasoning behind this thought being that the compression is specific to SCPS-TP headers and yet must be able to ride on top of normal IP traffic.[6] Thus, placing the compression within SCPS-TP satisfies both requirements. This also implies that the compression would be mostly done in software. Including it in hardware is unlikely, unless the hardware deals specifically with the entire SCPS protocol stack in which case it becomes cost inefficient. State is also kept for each active connection in the form of the last uncompressed TCP header for that connection. The decompressor uses the stored header to reconstruct incoming packets. Only the port information from the stored header is needed, as any other needed fields would be specified in the compressed packet.

4 IP Header Compression (RFC 2507)

The header compression scheme in [RFC2507] aims to satisfy several goals, including better response time, line efficiency, loss rate, and bit overhead. [RFC2507] is similar to [RFC1144] in regards to TCP, but includes support for other features and protocols, such as TCP options, ECN, IPv6, and UDP. The specification also allows extension to multicast, multi-access links, and other compression schemes which ride over UDP.

4.1 Compressed Header

Like most header compression schemes, [RFC2507] relies on many of the fields within a header to remain constant (or change seldomly) and thus, omitted from the compressed header. Other fields can be implied from information obtained elsewhere, such as the total packet length, which is inferred from the linker. Also, similar to [RFC1144] differences instead of absolute values can be used for certain TCP fields. Decompression is based on a connection identifier[7] and its associated context (stored information). Similar to [RFC1144] the CID is limited to 8 bits for TCP, but can be up to 16 bits for non-TCP connections, which is beneficial for multi-access links. Also, for non-TCP connections, each CID has a generation number, which is incremented each time the context changes. The generation value allows the decompressor to detect an outdated context while attempting to decompress packets.
Figure 8 shows the various different types of headers that can be compressed and the category in which each field falls, while Figure 7 explains each field.

[6] The SCPS documents call for such backwards compatibility.
[7] The connection identifier (CID) space is separate for TCP and non-TCP connections. Thus a TCP CID and a non-TCP CID with the same value do not identify the same context.

NOCHANGE - Fields which are expected to remain the same

INFERRED - Fields whose value are inferred from other values

RANDOM - Fields whose value changes unpredictably and must be included unmodified

DELTA - (Only if TCP is used) Fields whose change is reported as a difference from the previous packet

Figure 7: Field Activity Definitions [RFC2507]

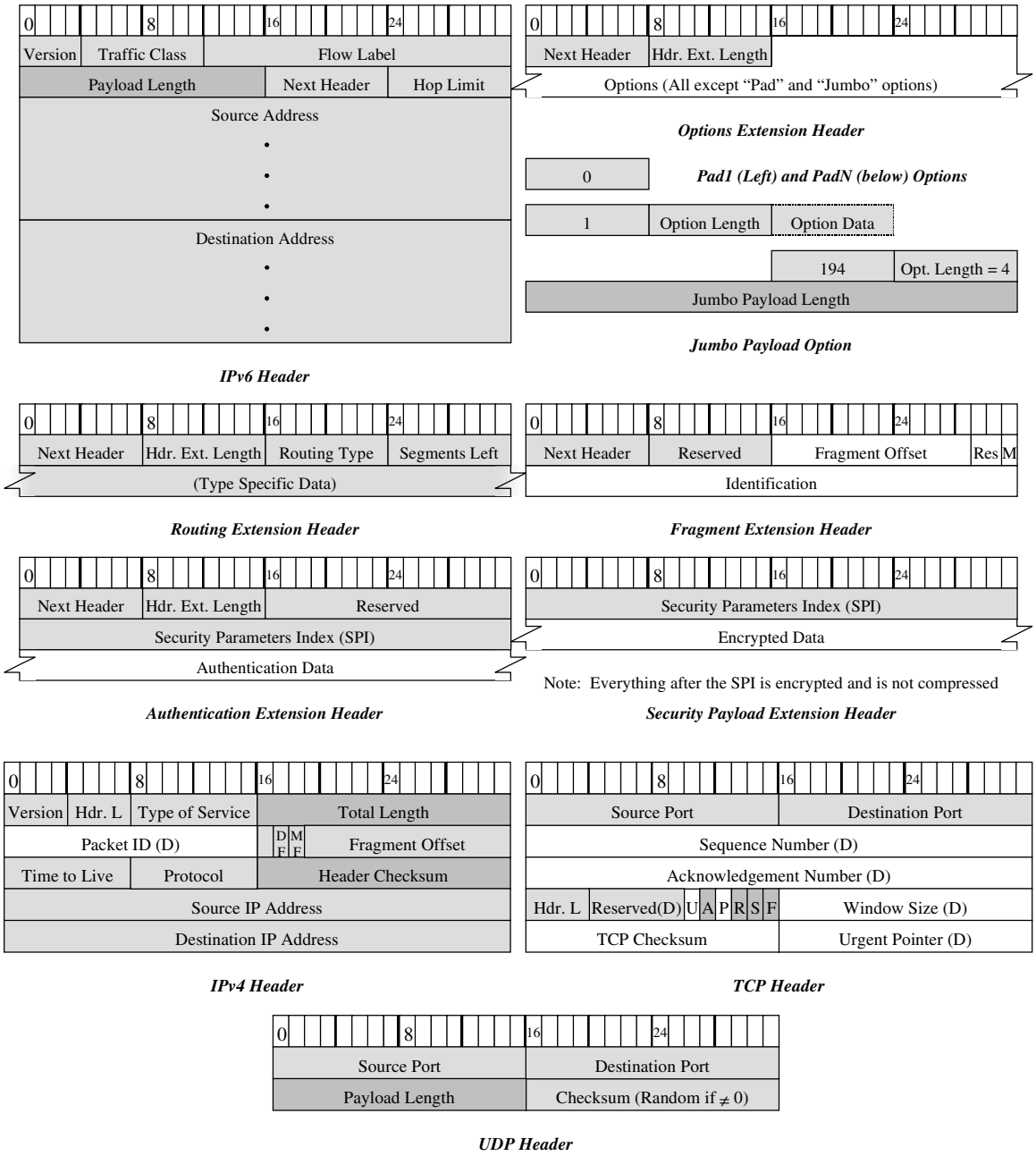

Figure 8: Activity of various protocol headers [RFC2507]

Fields marked as "NOCHANGE" are stored in the context and initialized or update by uncompressed packets. Since the fields are expected not to change, they are never sent in the compressed packet. If a change should occur, it forces an uncompressed packet to be

sent. Fields labeled as "RANDOM" are sent unmodified in the compressed header and must occur in the same order as the uncompressed header.[8] The formats of the compressed headers are shown in Figure 9. Connection identifiers are mandatory for compressed packets. For the TCP header, options and padding are transmitted uncompressed, only if the option has changed from the previous value. Options and padding for IPv4 are not supported. The "R-Octet" in the compressed TCP header allows for changes to the reserved field of the TCP header to be sent unmodified should it differ from that found in the context.[9] Thus, the R and O flags allow support for both TCP options and ECN.

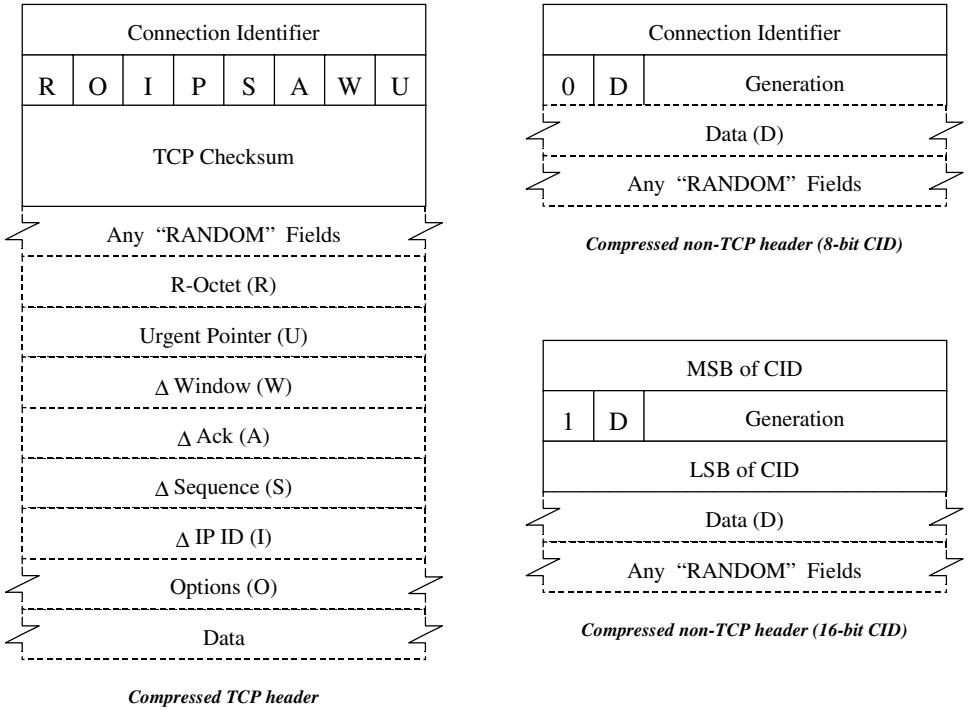

Figure 9: Compressed Header Formats [RFC2507]

4.2 Error Correction and Recovery

For TCP packets, the error recovery schemes specified in [RFC1144] are used, and an explanation of those schemes is described in section 2.2 of this document. In addition to these schemes, other methods can be used to attempt to recover packets which may fail a TCP checksum due to a previously missed delta, but otherwise contain valid data. For non-TCP packets, differential encoding is not used and the context is not damaged when loosing a packet. An uncompressed non-TCP packet which changes the context carries a

[8] Fields marked as "DELTA" may be sent as "RANDOM" (As is required for example of the Packet ID in IPv4 when not using TCP). There is a provision so that compressed TCP packets can be sent without deltas. The resulting compressed packet consists of the CID, Random Fields, and the whole TCP header (sans port information).

[9] Note that the "octet passed with the R flag MUST NOT update the context." [RFC2507]

different generation number as does all subsequent packets for that particular context. Should the uncompressed packet be lost, the generation number is not updated at the decompressor. Thus any subsequent compressed packets with the new generation number do not match that of the one stored at the decompressor and are dropped.
In order to prevent a large amount of data loss due to such a change in context an uncompressed packet is sent periodically, with the period backing of exponentially to a preset maximum period. Also, to protect against periods of disconnection, there is a maximum time between which uncompressed packets can be sent.
In order to offset the use of differential encoding over links with high loss characteristics, two approaches to quickly repair the context are offered. The first is entitled the "twice algorithm," which uses the assumption that deltas between compressed headers are often the same. Thus a compressed packet which failed the TCP checksum is decompressed again (delta's applied for a second time) and checked for correctness.[10] The second approach allows the decompressor to request a full header[11] from the compressor. The requests must be done on a limited bases, and a single feedback request may identify multiple contexts which need to be updated. Such header requests also require the presence of a two-way communication channel on the compressed link.

4.3 Compressor Location and Efficiency

The compressor is located between the network (IP) and framing layers, and relies on the framer to provide strong error detection as well in-order delivery. However, it is possible to support reordering through various mechanisms. For example, by adding a sequence number to compressed TCP packets, the decompressor can place packets in the correct order and preserve the corresponding deltas. Also, the framer must not pad UDP or tunneled packets and must be able to provide the decompressor with the total length of incoming packets. The cost of sending uncompressed packets at an exponentially increasing rate for non-TCP transfers is low, costing only a few bits to the average header size. State is kept at both the compressor and decompressor and consists of the CID, generation number (if applicable), and context for each connection. Also, there is no limit to the number of extension headers allowed in an IPv6 header, and so, the maximum size of the context is restricted to avoid excessive memory requirements. Thus, only the portion of a header that can fit within the maximum context is compressed.[12]

5 RTP Header Compression (RFC 2508)

The goal of [RFC2508] is to provide a means of reducing the cost of headers when using the Real-Time Transport Protocol (RTP), which is often used for applications such as audio and video transport. However, instead of compressing the RTP header alone, greater efficiency is obtained by compressing the combined RTP/UDP/IP headers together. Another important goal is that the implementations for the compression and decompression code need to be simple, as a single processor may need to handle many

[10] Various optimization tactics can be applied, such as trying increments of smaller deltas, but the basic concept remains the same. In the author's tests, between 83 and 99 percent of single data packet losses were successfully detected and corrected by this method. (53-99% for losses in the ACK stream)
[11] Requests are done via a CONTEXT_STATE packet, discussed in more detail in [RFC2507]
[12] More specifically, the compressor cannot partially compress a sub-header. Thus compression is applied to the initial sequence of whole sub-headers which is within the maximum.

interfaces. Finally, this compression scheme is not intended to work in conjunction with RTCP (Real-time Transport Control Protocol) as the required additional complexity is undesirable.

5.1 Compressed Header

The header compression scheme for RTP utilizes some of the same techniques for compression as those used for TCP. Figure 10 shows a standard RTP header. Many of the fields in the RTP header change by predictable amounts, and for certain fields, those differences often remain constant over a sequence of compressed packets. As a result,

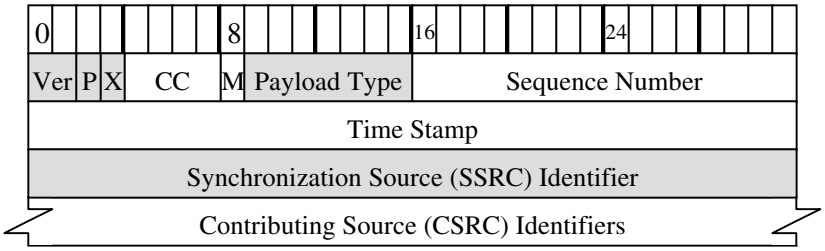

Figure 10: RTP Header [RFC1889]

the expected change can be implied without noting the actual difference in each compressed packet. [RFC2508] refers to these implied changes as "first-order" differences, and they are stored along with the context for each connection.[13] Fields using a delta encoding in the compressed headers indicate a change to the expected "first-

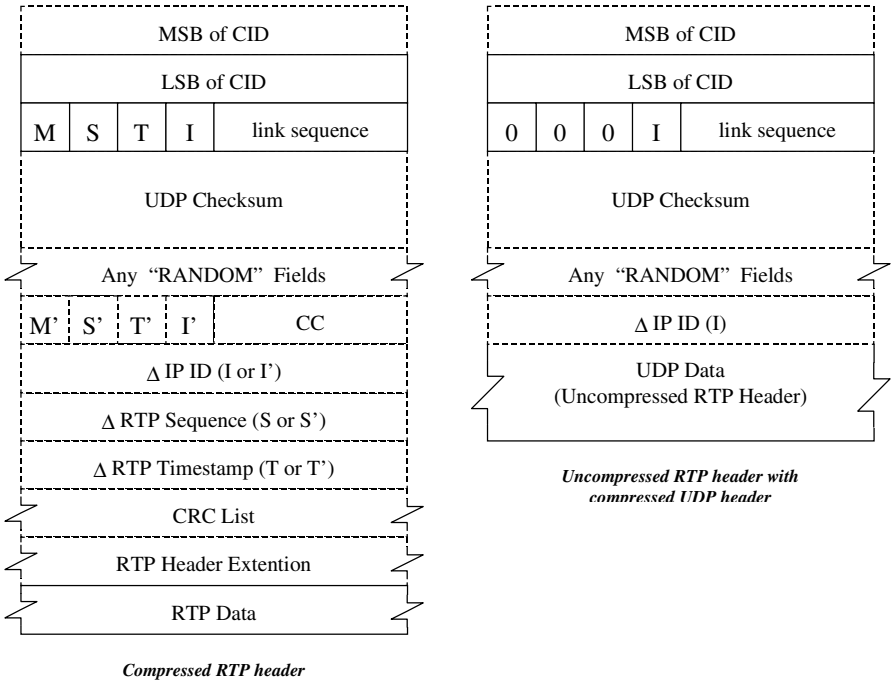

Figure 11: Compressed RTP Header Formats [RFC2508]

order" differences, and the amount of change is considered the "second-order" difference. Thus, if the "second-order" difference is not zero, the new "first-order" difference is

[13] First-order differences are set to zero whenever a uncompressed packet is sent or received.

transmitted in the corresponding field of the compressed header and the context is updated to reflect the change. Figure 11 shows the format for a compressed RTP/UDP/IP header. As a performance benefit, the UDP/IP header can still be compressed should an uncompressed RTP header need to be sent. The resulting header is also shown in the figure. As per [RFC2507] the compressed header allows for both 8 and 16 bit headers, and includes a four bit sequence number to aid in loss and reordering detection. The presence of the UDP checksum is stored in the context and is based on the presence of a nonzero checksum in the uncompressed header. Also, the M bit is an explicit copy of the M bit found in the uncompressed RTP header, and the S, T, and I bits indicate changes to the first order differences of their corresponding fields: sequence number, timestamp[14], and IP ID. Since changes to the CC count and CSRC list are typically the most infrequent, their inclusion is based on a special manipulation of the M, S, T, and I bits. When the bits are all set, the CC and CRC fields are present and the actual values for the bits are located in M', S', T', and I'. The inclusion of random fields are based of any random fields described in [RFC2507], which are also discussed in section 4.1 of this document. The delta fields in the compressed headers utilize a variable length encoding as apposed to a simple two's complement scheme. A default encoding is specified in [RFC2508] and the use of other encodings can be negotiated for each session, if appropriate and possible. Since different encodings can be optimal for various audio and video situations, common differences can be encoded as small values.

5.2 Error Correction and Recovery

It is possible to envision the use of RTP over both simplex and full-duplex links, and so both conditions need to be handled. The simplex case could be handled through periodic refreshes similar to the compression slow start scheme found in section 4.2. Such a scheme would also work in a full duplex scenario, but doing so sacrifices some advantages of having a full duplex connection. Using a header request system (again similar to that found in section 4.2) helps expediate recovery by notifying the compressor through the reverse channel. However, in conditions of high delay, the periodic refresh scheme is favorable. Generation numbers (section 4.2) are used to identify outdated contexts. Detection of loss and reordering is also assisted by the additional four bit sequence number within a compressed RTP header. The "twice algorithm", which is discussed in section 4.2, can be used to quickly recover from loss or reordering. In addition, by using the four bit sequence number the number of times to reapply the delta is know, although the risk for false positives increases with each iteration.

5.3 Compressor Location and Efficiency

Since this header compression scheme is an extended use of [RFC2507], the location, requirements, and efficiency of UDP transfers discussed in section 4.3 hold for [RFC2508] as well. However, extra memory requirements do exist, since the first-order differences, sequence number, and RTP header need to be stored in addition to the normal context of a UDP packet. Finally, the framing level no longer needs to provide in-order delivery as the four bit sequence number is used to detect reordering.

[14] The timestamp in the RTP header is not a traditional timestamp. Rather, they are time indicators for particular period samples in an audio stream or frame sample in a video stream. Thus a video frame that spans multiple packets would all carry the same timestamp.

6 Robust Header Compression (ROHC)

The robust header compression scheme [ROHC] is currently being developed in the IETF by a working group with the same name. The incentive for such a scheme arose from links with significant error rates, long round-trip times, and bandwidth limited capacity, and thus, the goal is to be able to design highly robust and efficient header compression schemes based upon a highly extensible framework. Consequently, the information presented in this document for this particular header compression scheme represents the underlying framework on which compression for other protocols is built. Finally, both UDP and RTP are covered in the [ROHC] document and are thus touched upon here as well.

6.1 Compressed Header

The ROHC framework, while more complicated than other techniques, shares the basic concept of storing common header fields for a particular packet stream. Compression and decompression are treated as finite state machines each of which is broken into three states. The compressor's three states include the initialization or complete refresh of the decompressor's context, the transmission of fully compressed headers, or the delivery of partially uncompressed data which may be needed to communicate unexpected changes or other irregularities. The decompressor's states are indications of the quality of the decompressor's context and can either be in-sync, confused, or invalid. An invalid state represents a context which must be updated or initialized.[15]

In addition to these states there are several modes in which a ROHC scheme can operate. The first is Unidirectional Mode (U-mode) which makes use of periodic refreshes and timeouts in order to keep the context current. All ROHC schemes start in U-mode and may transition to any of the two remaining modes upon reception of feedback from the decompressor. The second mode is Bidirectional Optimistic Mode (O-mode) and makes use of the feedback channel to send recovery requests.[16] The final mode, Bidirectional Reliable Mode (R-mode), utilizes the feedback channel to a greater degree than the previous two modes in order to best prevent loss of context synchronization. All context updates are acknowledged in this mode. Another feature of ROHC is the ability to use different encoding schemes for each header field. Since many fields often behave characteristically, the encoding schemes can be tailored to a particular field or set of fields - resulting in better compression. The encoding schemes are implementation specific, and a few example encoding methods are given in [ROHC].

The basic framework for a ROHC packet consists of padding, feedback information, header information, and remaining payload - in that order. Figure 12 shows the basic framework for a ROHC packet as well as the breakdowns of the four generic fields. The ordering allows the decompressor to quickly separate feedback information and pass it to the compressor. CID information is determined by the CID space for a given channel. If the CID space is small (values of 0-15) then the "Add-CID" octet is used, otherwise the

[15] The [ROHC] document lists the states as follows:
For the compressor: Initialize and Refresh (IR) ↔ First Order (FO) ↔ Second Order (SO)
For the decompressor: No Context (NO) ↔ Static Context (SC) ↔ Full Context (FC)

[16] Acknowledgements for significant context updates may also be sent.

"CID" field is used.[17] The IR/IR-DYN format is used to update a context, either fully (IR) or partially (IR-DYN). An IR packet must be used to initialize the decompressor, while a IR-DYN is typically used to update more "dynamic" portions of the context.[18]

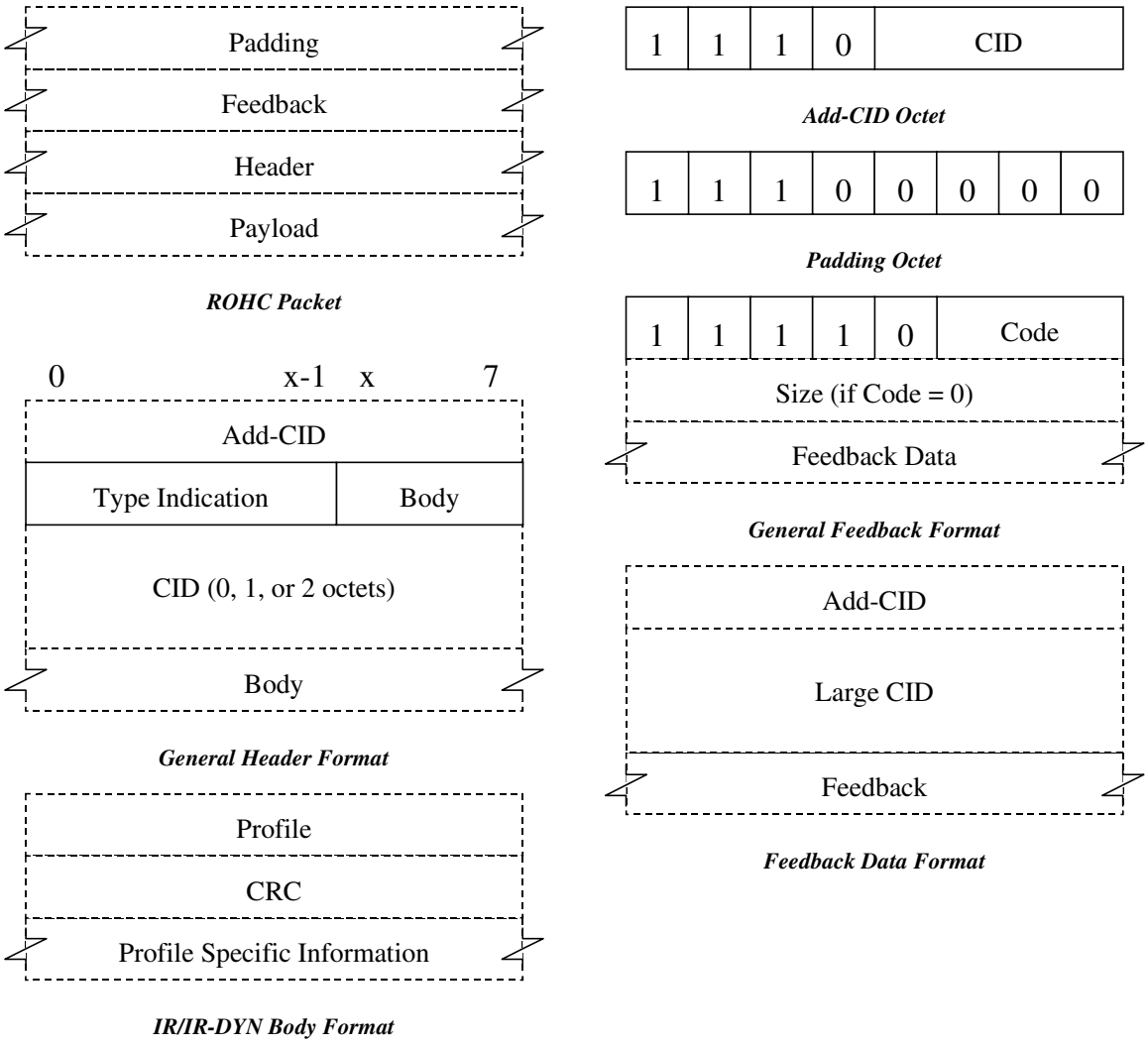

Figure 12: ROHC Framework

Figure 13 and 14 shows the classification of various header fields as well as sample compressed header fields and formats for an sample implementation of a RTP header compression scheme detailed further in [ROHC]. In Figure 13, fields marked as know are never sent in a compressed header as they are always presumed to have a specified value. The remaining field definitions are similar to those described in section 4.1.

[17] A CID value of zero is always omitted from the header. Large CIDs are encoded with self-describing variable-length values.

[18] Both IR and IR-DYN formats can be used to update part of the context. Also, the two formats are distinguished by their type identification of $(1111110\text{-xbit})_2$ for IR and $(11111000)_2$ for IR-DYN.

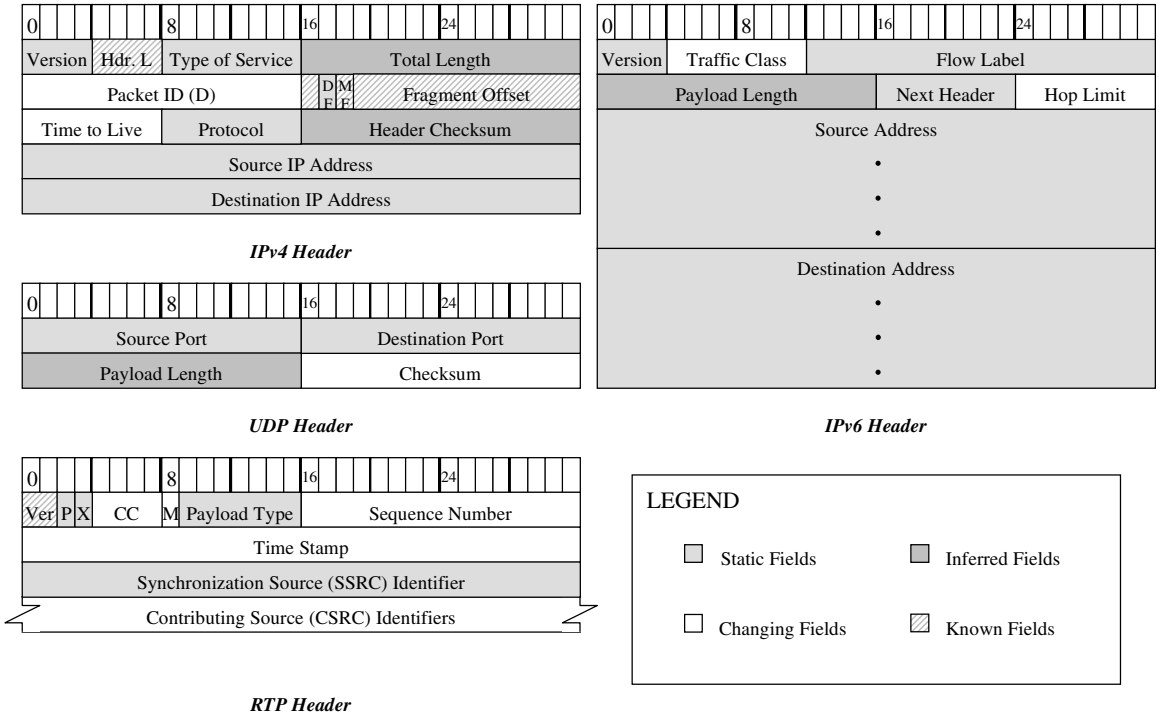

Figure 13: Activity or various protocol headers [ROHC]

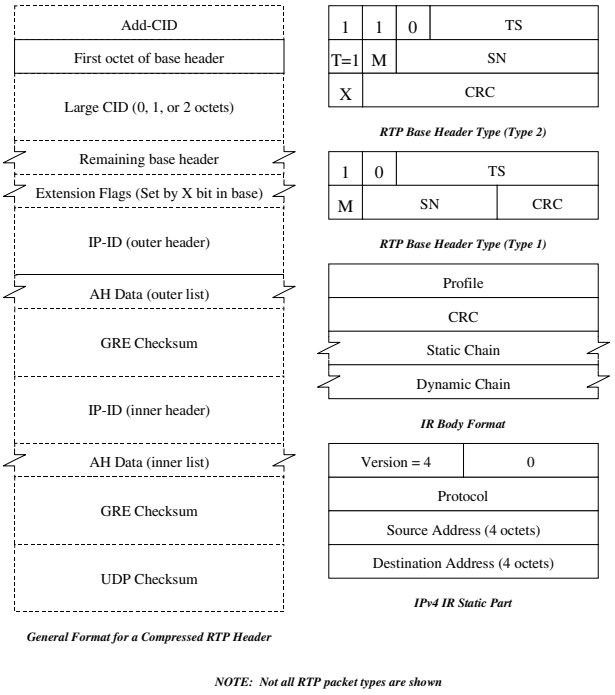

Figure 14: Select headers of a sample implementation of RTP ROHC [ROHC]

6.2 Error Correction and Recovery

Since the bit error rate (BER) is non-trivial[19] for ROHC's intended application set, it is important for the framework to be able to handle errors occurring over the compressed link. CRC's and sanity checks are used in addition to any error checking done at lower levels. As noted in the section above, CRC's are carried in the IR and IR-DYN headers in order to prevent the delivery of damaged data to the upper layers, due to a corrupted context update. Depending on which mode of operation ROHC is in, feedback can be used to acknowledge changes (ACKs), indicate loss (NACKs), or request context (STATIC-NACKs). Feedback can also be obtained by other means, such as from the lower layers themselves, but such feedback is implementation specific. Periodic refreshes similar to the connection slow start described in section 4.2 are only performed when ROHC is in the unidirectional mode. Periodic refreshes lead to a greater probability of loss propagation, but also allows ROHC to be used in situations where a return path over a link is unavailable. ROHC also protects fragmented packets[20] by using a CRC which covers the entire segment prior to fragmentation.

6.3 Compressor Location and Efficiency

Like nearly all header compression schemes, ROHC takes place at the link level of the network stack. The linker is required to provide in order delivery, without duplication, but does not necessarily need to provide strong error protection.[21] Also, the ROHC scheme is rather complex. However, as is often the case with cellular links, bandwidth is very costly, especially when compared to the cost of processing power. Thus, the robustness and efficiency of the header compression scheme is more of a concern than its simplicity.

7 Failure of Header Compression with Security

Many of the common header compression schemes lose their effectiveness when security measures are employed. For example, IP Security (IPSec) is commonly used and encrypts data and headers for a given packet, save for the authentication fields and select IP fields needed for routing. Unfortunately, one of the main reasons that allows header compression to be possible is the ability to extract the significant redundancy between header fields for a particular packet stream. IPSec encrypts information necessary to identify packets for a particular stream, and so a context cannot be tied to a particular packet stream. Furthermore, many of the headers involved in the compression process are encrypted prior to its arrival at the compressor, since many compressors lie below the network layer. Thus, changes in key fields would not be detected, due to the encryption. These reasons alone are enough to discount the effectiveness of header compression for secured data transfers.

[19] Bit error rates can reach as high as $1e^{-2}$

[20] Identified by a type identification field of $(1111111\text{-fbit})_2$ where the f-bit is used to indicate the last fragment.

[21] Although, strong error protection on the part of the linker is recommended.

8 Summary

The following section provides a summary of the different header compression techniques discussed in this document.

Scheme	Supports	Header Size (bytes)		Additional Notes
		Typical Compressed	Orig.[22]	
RFC 1144	TCP/IPv4	(4-7)	(40)	
SCPS	SCPS-TP	SCPS-NP (4-16) SCPS-TP (4-14)	(44+)	Specifically tailored for a SCPS environment.
RFC 2507	TCP, UDP, IPv4, IPv6, IPv6 Extension Headers, Options, ECN	TCP (4-7) or (6-9) with second seq. number for reordering, TCP with no deltas (17), Non-TCP (2-5)	TCP w/IPv4 (40+), w/IPv6 (60+)	Support for RTP and other schemes riding over UDP via hooks. Extensible to multicast and multi-access links.
RFC 2508	RTP/UDP/IPv4	(2-4), 2 without UDP checksums, 4 with UDP checksums	(44+)	Designed to work with a general compression framework as in [RFC 2507] allowing for IPv6 and IPv4 support.
ROHC	RTP, ESP, UDP, IPv4, IPv6, others	Implementation Specific, Framework (1-4), Sample RTP (1-6)	Vari-able	Designed to be highly extensible, such that adding schemes for additional protocols is simple.

Table 2: Summary of Header Compression Schemes

References

[RFC1144] Van Jacobson, “Compressing TCP/IP Headers”, RFC 1144, February 1990

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, RFC 1889, January 1996

[RFC2460] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460, December 1998

[RFC2507] M. Degermark, B. Nordgren, and S. Pink, “IP Header Compression”, RFC 2507, February 1999

[RFC2508] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999

[RFC791] Jon Postel, “Internet Protocol”, RFC 791, September 1981

[RFC793] Jon Postel, “Transmission Control Protocol”, RFC 793, September 1981

[ROHC] C. Burmeister, M. Degermark, H. Fukushima, et. All, “RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", Internet Draft <draft-ietf-rohc-rtp-09.txt>, February 2001

[SCPS-NP] Space Communications Protocol Specification (SCPS)—Network Protocol (SCPS-NP). Recommendation for Space Data System Standards, CCSDS 713.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1999.

[SCPS-TP] Space Communications Protocol Specification (SCPS)—Transport Protocol (SCPS-TP). Recommendation for Space Data System Standards, CCSDS 714.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1999.

[22] The original header size is dependent on the combination of protocols used. Thus, depending on the particular header compression scheme, there may be one or more original header size, and all permutations may not be listed. Typical minimum header sizes include: TCP (20), TCP with timestamps (32), UDP (8), IPv4 (20), IPv6 (40), and RTP (12). The header size for a set of protocols is calculated by summing each appropriate value.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2001	Technical Memorandum

4. TITLE AND SUBTITLE

Survey of Header Compression Techniques

5. FUNDING NUMBERS

WU–258–90–00–00

6. AUTHOR(S)

Joseph Ishac

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–13010

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM—2001-211154

11. SUPPLEMENTARY NOTES

Responsible person, Joseph Ishac, organization code 5610, 216–433–6587.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 17, 61, and 62 Distribution: Nonstandard

Available electronically at http://gltrs.grc.nasa.gov/GLTRS

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This report provides a summary of several different header compression techniques. The different techniques included are: (1) Van Jacobson's header compression [RFC1144]; (2) SCPS header compression [SCPS-TP, SCPS-NP]; (3) Robust header compression [ROHC]; and (4) The header compression techniques in [RFC2507] and [RFC2508]. The methodology for compression and error correction for these schemes are described in the remainder of this document. All of the header compression schemes support compression over simplex links, provided that the end receiver has some means of sending data back to the sender. However, if that return path does not exist, then neither Van Jacobson's nor SCPS can be used, since both rely on TCP. In addition, under link conditions of low delay and low error, all of the schemes perform as expected. However, based on the methodology of the schemes, each scheme is likely to behave differently as conditions degrade. Van Jacobson's header compression relies heavily on the TCP retransmission timer and would suffer an increase in loss propagation should the link possess a high delay and/or bit error rate (BER). The SCPS header compression scheme protects against high delay environments by avoiding delta encoding between packets. Thus, loss propagation is avoided. However, SCPS is still affected by an increased BER since the lack of delta encoding results in larger header sizes. Next, the schemes found in [RFC2507] and [RFC2508] perform well for non-TCP connections in poor conditions. [RFC2507] performance with TCP connections is improved by various techniques over Van Jacobson's, but still suffers a performance hit with poor link properties. Also, [RFC2507] offers the ability to send TCP data without delta encoding, similar to what SCPS offers. ROHC is similar to the previous two schemes, but adds additional CRC's into headers and improves compression schemes which provide better tolerances in conditions with a high BER.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Header; Compression; Network	26
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	